CENTRE DE RECHERCHE
SUR LES TRANSPORTS
(C.R.T.)
CENTRE FOR RESEARCH
ON TRANSPORTATION

UNIVERSITÉ DE MONTRÉAL
ÉCOLE DES HAUTES ÉTUDES COMMERCIALES
ÉCOLE POLYTECHNIQUE

CENTRE DE RECHERCHE SUR LES TRANSPORTS
UNIVERSITÉ DE MONTRÉAL
C.P. 6128, SUCCURSALE A,
MONTRÉAL
CANADA H3C 3J7

TÉLÉPHONE: (514) 343-7575
TÉLEX: (21) 05-5613 '5
EMAIL: crt@crt.umontreal.ca
TÉLÉCOPIEUR (FAX): (514) 343-7121

92-04605/

CRT-816

Simulating Vehicle Dispatching with Linear Programming Models

by

Jean-Yves Potvin
Gina Dufour
Jean-Marc Rousseau

Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succursale A, Montréal, Québec, Canada H3C 3J7

Centre de recherche sur les transports — Publication #816

February 1992

Dépôt légal – 1er trimestre 1992

Résumé. Dans cet article, nous expliquons comment on peut simuler le processus de choix multicritères d'un répartiteur de véhicules à l'aide de la programmation linéaire. Nous introduisons en particulier une méthode d'apprentissage automatique permettant au système informatique de s'adapter au processus de pensée de l'expert par un ajustement progressif des facteurs de pondération liés à chacun des critères de décision. Des résultats expérimentaux dans différents contextes de décision sont également présentés.

Mots Clés. Choix Multicritères, Programmation Linéaire, Répartition de Véhicules, Apprentissage.

Abstract. In this paper, we explain how the multiattribute choice problem of expert vehicle dispatchers can be solved with linear programming models. We introduce in particular an incremental weight learning scheme that allows a computer system to dynamically learn the dispatcher's decision process as new demands come in and new decisions are taken. Results in various dispatching contexts are reported.

KeyWords. Multiattribute Choice, Linear Programming, Vehicle Dispatching, Learning.

Scope and Purpose. Vehicle dispatching is aimed at appropriately selecting among a fleet of vehicles, the one that will be assigned to the service of an incoming request. Up to now, this decision problem has been mainly solved by human experts. We thus report here about ways to build a computer system that can learn to automatically simulate the decision process of an expert dispatcher based on examples of his(her) previous decisions.

1. INTRODUCTION

Vehicle dispatching is aimed at efficiently allocating vehicles for servicing real time requests from customers. Typical examples are ambulance services, parcel pick-up and delivery services and priority mail. Vehicle dispatching is still mainly in the realm of human expertise because it requires a lot of experience, judgement and expertise. For example, when a new request comes in, the dispatcher must have knowledge of the following facts before taking a decision (that is, before allocating a specific vehicle to a new request):

- current location of each available vehicle
- planned route and schedule of each available vehicle (for previously allocated requests).
- location of the pick-up and delivery point of the new request
- distances and travel times
- specific characteristics of the underlying transportation network

Dispatchers are thus a key element in any organization providing that type of transportation service. Unfortunately, it is not easy to find such people, because specific skills are required for adequately performing the dispatching task. Moreover, it takes a long time to train these persons and their professional carreer typically lasts only a few years because of the high stress associated with reacting properly and quickly to a dynamically evolving environment.

Hence, builing an expert system that would act as a professional dispatcher would certainly be very useful. On the other hand, our own experience has shown that it is very difficult, if not impossible, to formalize the knowledge of such dispatchers (e.g. as decision rules). We thus turned ourselves towards automated learning techniques, in particular learning by examples, in order to learn to reproduce the decision behavior of a dispatcher, based on examples of previous decisions. It is much more easy to get such data (since taking decisions is the usual task of the dispatcher) than to extract explicit decision rules from a dispatcher.

There are many different techniques to automate learning. In particular, neural networks [Rumelhart and McClelland 86] and decision trees [Quinlan 86] are very well known techniques. In this paper, we rather hypothesize that the dispatching task can be modeled as a multiattribute choice problem and develop an approach based on linear programming in order to learn to appropriately weight the various attributes.

In the next section, we first describe more precisely the vehicle dispatching problem. Then, we present the problem as a multiattribute choice problem and describe linear programming models aimed at properly weighting the various attributes. Finally, we explain how to incrementally build and solve such linear programs so as to dynamically learn and adjust the weights, as the dispatcher takes new decisions.

2. THE VEHICLE DISPATCHING PROBLEM

In the Vehicle Dispatching or Dial-A-Ride Problem [Bodin et al. 83], customers call a dispatcher for service request. Each customer specifies a distinct pick-up and delivery location and, perhaps, a desired time for pick-up or delivery. If all customers demand immediate service, then routing and scheduling is done in real time and the problem is referred to as the dynamic Vehicle Dispatching Problem. In such a demand responsive system, the system is looked at when a new customer calls into the dispatch office. At that time, some of the earlier customers have been delivered to their destinations and, hence, are no longer considered. The remaining customers who have requested service have been assigned to a vehicle and are either on their way to their destination or are waiting for their pick-up. Moreover, at that time, the route and schedule for each vehicle is known. The problem is thus to determine the assignment of the new customer to a vehicle and the new route and schedule for the vehicle that the customer is assigned to.

Currently, this problem is still mainly solved by human dispatchers. Only a few algorithmic approaches have been designed for that class of problems, the best known approach being Wilson's heuristic [Wilson and Covin 79] for a dial-a-ride transportation system in Rochester, NY. A dynamic programming algorithm for the single vehicle case is also described in [Psaraftis 80].

Our approach is quite different, since we are interested in building an adaptive system that can learn to reproduce the decision process of the expert dispatcher it is trained with. In this way, such a system could easily adapt to various dispatching environments. Since the decision procedure can vary a lot from one organization to another, the adaptability of the system looks here as a great asset.

In order to model that problem as a multiattribute choice problem, let us first assume that each one of the n available vehicles (drivers, candidates, alternatives) involved in the dispatching situation can be characterized by a vector of m attribute values $(V_{j1}, V_{j2},...,V_{jm})$, $j=1,2,...,n$. Those attributes should reflect, for each vehicle, the actual dispatching situation with respect to their current route and the new request to be serviced, like: the extra-mileage required for servicing the new request, the pick-up time, the delivery time, additional delays incurred by the requests that are planned to be serviced by that vehicle, etc.

The set of all candidates, each with its m attribute values, is called a *context* for decision making. The objective of the dispatching process is thus, given a context, to select a suitable vehicle. This is a typical multiattribute decision problem. We thus hypothesize here that the mental model of the dispatcher fits that model and that he(she) is really weighting a multicriteria utility function in order to take good decisions.

We will explain more precisely in the next section some basic assumptions and simplifications with respect to our multiattribute choice model. However, it should be noted right now that the dispatching expertise is basically encoded within the set of attributes for describing a dispatching situation and secondly within the weights that are assigned to each attribute. Here, the set of attributes for properly describing a dispatching situation is set a priori (and has not to be discovered!). The aim of the system is thus to discover the weighting of those a priori attributes in order to reproduce the dispatcher's decision behavior.

3. THE MULTIATTRIBUTE CHOICE PROBLEM

In the following, we note the set of *n* alternatives (vehicles, drivers) as A and the set of *m* criteria or attributes for describing a dispatching situation for a given vehicle as C. We note by V_{jp} the value of alternative j for attribute p, $j \in A$ and $p \in C$.

As described in previous papers [Lynch 85, Oral and Kettani 89], three main steps take place within the multiattribute choice process:

- *Valuation stage.* Given the value of alternative j for attribute p, V_{jp}, the decision maker assigns a subjective non-negative number d_{jp} to it. In other words, the decision maker uses a mental valuation function (partial utility function) f_p such that $d_{jp} = f_p(V_{jp})$, for all j and p. In this paper, it is assumed that $d_{jp} = f_p(V_{jp}) = w_p V_{jp}$ where w_p is the weight given to attribute p.

- *Integration stage.* The decision maker then combines the partial utilities d_{jp}, $p \in C$ so as to produce an integrated overall utility U_j for each alternative $j \in A$. Here, models with additive forms are used, that is:

$$U_j = \Sigma_p \, d_{jp} = \Sigma_p \, w_p V_{jp}.$$

- *Conclusion stage.* Finally, the subjective overall utilities are transformed into an order-ranking or a set of pairwise preferences. In our case, since the dispatcher expresses a preference for one vehicle over all other available vehicles, we use pairwise preferences. Assuming that n vehicles {1,2,..., n} are available and that vehicle i* is selected for servicing the new request the set of (n-1) pairwise preferences P would be the following:

$$P = \{(i^*, i),\ i = 1,...,n \text{ and } i \neq i^*\}\ .$$

In other words, vehicle i* is preferred over all other vehicles for servicing the request.

Two different types of techniques can be applied to our choice problem: mathematical programming methods [Shocker and Srinivasan 79; Mustafi and Xavier 85; Jacquet-Lagrèze and Siskos 82] and statistical methods [Green and Rao 71; Beckwith and Lehmann 73]. Recent literature shows that the estimates obtained from linear programming models show greater stability and better predictive ability [Srinivasan and Shocker 73a,b; Horsky and Rao 84; Jain et al. 79; Wittink and Cattin 81]. In the following, we thus describe a mathematical programming model for estimating the weight of each attribute and simulate the dispatcher's decision behavior in the best possible way.

4. THE LINEAR PROGRAMMING MODEL

Based on the values V_{jp}, $j \in A$, $p \in C$, and a set of preferences P observable from a dispatcher's decision, the following linear programming model, known as the Srinivasan and Shocker model [Srinivasan and Shocker 73a,b], can be used for estimating the relative weight w_p of each attribute p:

$$(3.1)\quad \text{Min} \sum_{(j,k)\in P} Z_{jk}$$

subject to:

$$(1)\qquad \sum_p w_p(V_{jp}-V_{kp}) + Z_{jk} \geq 0, \text{ for all } (j,k) \in P$$

$$(2)\qquad \sum_p w_p = \text{Constant } (> 0)$$

$$(3)\qquad w_p \geq 0, \text{ for all } p \in C$$
$$Z_{jk} \geq 0, \text{ for all } (j,k) \in P$$

Here, the weights to be estimated are the variables w_p. The variables Z_{jk} are indicating the degree of deviation of the model from the preferences expressed by the expert. If the objective value is equal to 0 at the optimum (i.e. all the Z_{jk}'s are equal to zero), then the corresponding set of weights w_p exactly reproduces the preferences in the set P, since:

$$\sum_p w_p(V_{jp}-V_{kp}) \geq 0, \text{ for all } (j,k) \in P$$

$$\Leftrightarrow$$

$$U_j - U_k \geq 0, \text{ for all } (j,k) \in P,$$

If the objective value is greater than zero, then there are one or more positive Z_{jk}'s, and the corresponding preferences are not satisfied. Hence, the global utility of the vehicle selected by the dispatcher is not the highest among all other available vehicles, and the model does not correctly reproduce the decision problem. Constraint (2) is simply aimed at eliminating the trivial solution $w_p = 0$, for all p.

As pointed out in [Oral and Kettani 89], such a model has two major weaknesses:

(a) partial utility functions are always linear functions of attribute values.

(b) the weighting is independent of the values V_{jp}. This is undesirable if, for example, the marginal utility should increase as the value of a given attribute increases.

In order to overcome such weaknesses and introduce non-linearity into the model, the various weights will be modeled as step functions of the attribute values. In the following, we describe a new enhanced linear program, as proposed initially in [Oral and Kettani 89].

We first assume that the range of all possible values of attribute p is divided into non-overlapping and exhaustive smaller intervals $E_{ip}=[L_{ip},L_{i+1,p}]$, $i=1,2,...,I_p-1$, such that the weight w_{ip} for interval E_{ip} is unique, for all i (note however that there might be more than one interval with the same weight value). We also assume that the partial utility function of each attribute p is non-decreasing. The model can now be described as follows:

(3.2) Min $\Sigma_{(j,k)\in P} Z_{jk} + \Sigma_p \Sigma_{i=1,\ldots,I_p-1} (Y^-_{ip} + Y^+_{ip})$

subject to:

(1) $\Sigma_p (w_{i_jp} C_{i_jp} - w_{i_kp} C_{i_kp}) + Z_{jk} \geq 0$, for all $(j,k) \in P$

with $V_{jp} \in E_{i_jp}$, $V_{kp} \in E_{i_kp}$
C_{i_jp} center of interval E_{i_jp}
C_{i_kp} center of interval E_{i_kp}

(2) $w_{ip} C_{ip} \leq C_{(i+1)p} w_{(i+1)p}$, for all $i=1,2,\ldots,I_p-1$ and $p \in C$

(3) $w_{ip} - w_{(i+1)p} + Y^-_{ip} - Y^+_{ip} = 0$

(4) $\Sigma_p \Sigma_{i=1,\ldots,I_p-1} w_{ip} =$ Constant (> 0)

(5) $w_{ip} \geq 0$, for all $i=1,2,\ldots,I_p-1$, $p \in C$
$Z_{jk} \geq 0$, for all $(j,k) \in P$
$Y^-_{ip}, Y^+_{ip} \geq 0$, for all $i=1,2,\ldots,I_p-1$, $p \in C$

The interpretation of the variables Z_{jk} is the same as in the first model and the first summation in the objective function is thus aimed at minimizing the deviations from observed preferences. The second summation, involving the new variables Y^-_{ip} and Y^+_{ip}, is aimed at minimizing the number of distinct weights w_{ip} for any given $p \in C$ (and thus eliminating unnecessary intervals that are included in the initial formulation of the problem). This interpretation is clear by looking at constraint (3). Finally, constraint (2) assures that partial utilities are non-decreasing and constraint (4) forbids the trivial solution.

The main difference between this model and the model proposed by Oral and Kettani stems from the fact that we do not take the real values V_{jp} in constraint (2), but rather the center of the intervals E_{ijp}, $V_{jp} \in E_{ijp}$. In the model of Oral and Kettani, it is implicitly assumed that there is exactly one value V_{jp} in each interval E_{ijp}, an assumption that does not hold in our case. We thus hypothesize that all the values within an interval can be considered as equivalent, and we replace

those values by the center of the interval in the linear programming formulation.

5. AN INCREMENTAL WEIGHT LEARNING SCHEME

Since a dispatcher assigns a new request to a single vehicle among n alternative vehicles, each decision adds (n-1) pairwise preferences in the set P and, consequently, (n-1) new preference constraints in the linear program (3.2). To correctly estimate the weights, we obviously need many such examples of decisions, and the linear program can thus quickly become very large (note, however, that the number of constraints of type (2), (3) and (4) does not increase with the number of examples).

In order to reduce the size of the linear programs to be solved (and, at the same time, reduce the computation time), we propose an approach where the weights are incrementally updated through a sequence of smaller problems.

This dynamic approach is summarized here:

1. Build the linear program (3.2) with the first decision example. Then, solve it to get an initial weight estimate.

2. As long as decision examples remain, do the following:

   Take the pairwise preference constraints of the current example that are satisfied by the current weights, and add them to set S.

   Take the pairwise preference constraints of the current example that are not satisfied by the current weights, add them to the linear program and solve the new system to get a new set of weights.

At the end, when all decision examples have been processed, do the following:

3. As long as pairwise preferences in set S are not satisfied by the current weights:

   Remove them from set S and add them to the linear program. Solve the system to get a new set of weights.

Such an incremental updating scheme allows a drastic reduction in the size of the linear programs to be solved (see section 6) and the weighting can thus be computed even if the number of examples is very large (note that M examples with n alternative vehicles in each example generate M*(n-1) pairwise preference constraints). The number of constraints being typically much larger than the number of weights to be estimated, the incremental approach in fact discards redundant constraints from the linear program.

The next section now presents some experimental results obtained with the linear programming model. In the last experiment, comparisons are provided between the incremental weighting scheme as described here and the global approach where a single linear program is solved only once to get the final set of weights.

6. COMPUTATIONAL RESULTS

In order to perform our experiments and collect the appropriate data, we used a dispatching decision support system called RAO (for Répartition Assistée par Ordinateur, see [Rousseau and Roy 88]). This system includes features to automatically generate a new request, to locate the customer in the underlying transportation network based on his(her) address and to compute useful informations for the assignment of the request to an appropriate vehicle. However, RAO does not suggest any decision to the dispatcher. It is up to him(her) to assign the new request to an appropriate vehicle based on informations and statistics provided by the computer system.

In the following, we first introduce the list of attributes aimed at describing a dispatching situation for any given vehicle. We then define the three distinct experimental contexts for testing the linear programming approach. Results are reported and the weighting schemes, as derived from the model, are then presented. Finally, comparisons are made between the global approach, which is solving a single large linear program, and the incremental weight learning approach, which is solving a sequence of smaller linear programs (cf section 5).

6.1 Description of dispatching situations

Among the informations provided by RAO, we selected the following attributes to characterize a dispatching situation for any given vehicle:

(1) *Minimum extra-mileage* (expressed in time units). RAO puts the new request at the minimum extra-mileage location within the route of the vehicle and derives the other measures from that location.

(2) *Pick-up time*

(3) *Delivery time*

(4) *Delay over 30 minutes for picking-up the new request.* This attribute is aimed at introducing some service quality considerations into the selection of an appropriate vehicle. Here, a delay over 30 minutes between the call from the customer and the pick-up is considered undesirable.

(5) *Delay over 90 minutes for delivering the new request..* This attribute is similar to the one described in (4).

(6) *Average of additional delays for picking-up the requests already included into the vehicle's route.* When a new request is included within a route, it can potentially introduce additional delays to the customers that are planned to be serviced by that vehicle (unless it is inserted at the end of the route). This attribute is thus aimed at evaluating that effect.

(7) *Number of additional delays for picking-up the requests already included into the vehicle's route.* Although the average of additional delays provides some useful informations, it is also important to know the number of such delays (e.g. to choose between a large average delay at few customers and a small average delay at many customers).

(8) *Average of additional delays for delivering the requests already included into the vehicle's route.* This attribute is similar to the one described in (6).

(9) *Number of additional delays for delivering the requests already included into the vehicle's route.* This attribute is similar to the one described in (7).

(10) *Current number of customers in the route of the vehicle.* Usually, it is good practice to equally share the workload among the vehicles. Hence, a vehicle is more likely to be selected if the

current number of customers serviced by that vehicle is small with respect to the other available vehicles (and conversely).

It should be noted that all these attributes are "cost" measures in the sense that a vehicle is less likely to be chosen when those values are high. In order to get monotonically increasing functions, we thus worked with "non utility" functions and values rather than utility functions and values.

For the experiments, we divided the range of values of each attribute in 10 intervals, ranging from 0 to some pre-specified limit (distinct for each attribute). In each case, an additional interval was created for grouping all the values beyond the limit, for a total of 11 intervals. The limits were set in such a way that a vehicle with any attribute value over its pre-specified limit would almost certainly be ignored by the dispatcher.

6.2 Experimental Contexts

We designed three different experiments which were then run on a Sun3 workstation using the XLP software [Marsten 82]. In each case, we gathered the data from an operation day of a real courrier service company. 100 dispatching situations were used for computing the set of weights (referred to as the "training examples") and 100 other situations for testing the performance of the system with those weights (referred to as the "testing examples"). 10 alternative vehicles were involved in each dispatching situation. The first two experimental contexts were based on simple dispatching rules and only the first three attributes were selected for describing a dispatching situation (i.e extra-mileage, pick-up time, delivery time). We felt that it would be useless to try to simulate the decision process of a human being if the linear programming model would fail with simple systematic decision rules based on a restricted set of attributes.

In the last experiment, we used the ten attributes introduced in section 6.1 and asked an expert at Centre de Recherche sur les Tranports (CRT) of Montreal University to take the dispatching decisions based on such data (both for the training and testing examples).

We summarize below the three distinct experimental contexts:

Experiment 1. The first dispatching rule that we want to reproduce is the following. First, we identify a subset of qualified

vehicles, by considering all vehicles with a pick-up time value within 5 minutes of the best value (i.e. all vehicles with a pick-up time in the "neighborhood" of the best value). Then, for all vehicles satisfying the first condition, we select the vehicle with minimum delivery time. In case of equality, all vehicles with the same minimum delivery time are selected.

Experiment 2. The second dispatching rule is more complex. As in the first rule, we identify first an initial subset of qualified vehicles, by considering all vehicles with a pick-up time within 5 minutes of the best value. Then, we filter again that subset by considering all vehicles in the subset with an extra-mileage value within 5 minutes of the best extra-mileage value. Finally, among the remaining vehicles, we select the one with minimum delivery time. In case of equality, all vehicles with the same minimum delivery time are selected.

Experiment 3. Here, a human expert took the decisions and only one vehicle was selected by the expert in each dispatching situation.

6.3 Computational Results

All the results and weighting schemes presented in this section were obtained by solving a single large linear program with 100*[10-1] = 900 preference constraints (with 100 standing for the number of dispatching situations and 10 for the number of alternative vehicles in each situation). We will thereafter refer to that methodology as the "global approach".

The predictive power of the model was then estimated by applying the weighting scheme to a distinct set of dispatching situations and by comparing the vehicle with highest utility (lowest non utility) with the vehicle selected by the expert or the rule. Note that the utility value (non utility value) of a given vehicle with respect to some attribute $p \in C$ was evaluated by linear interpolation between the levels $w_{ip}C_{ip}$ and $w_{(i+1)p}C_{(i+1)p}$, $i=1,2,...,I_p-1$. In that way, we get monotonically increasing functions f_p (as opposed to functions defined as $w_{ip}V_{jp}$ for any $V_{jp} \in E_{ip}$, see figures 1 and 2).

We must also observe that we values V_{jp} were normalized in order to get weights with similar magnitude. This was done by dividing each value V_{jp} by the pre-specified limit of attribute p. If the

value V_{jp} was over the limit, the normalized value was simply set to 1.

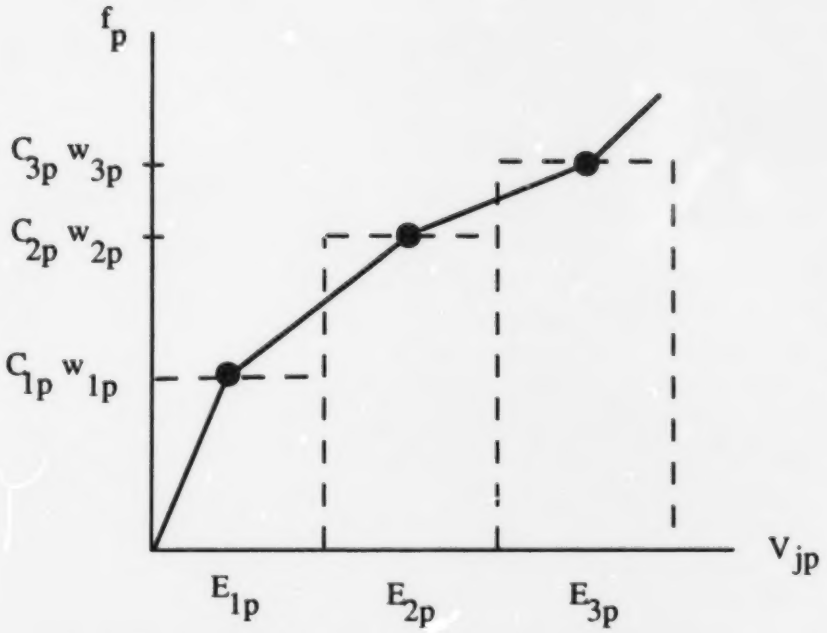

Figure 1 Monotonically increasing function f_p

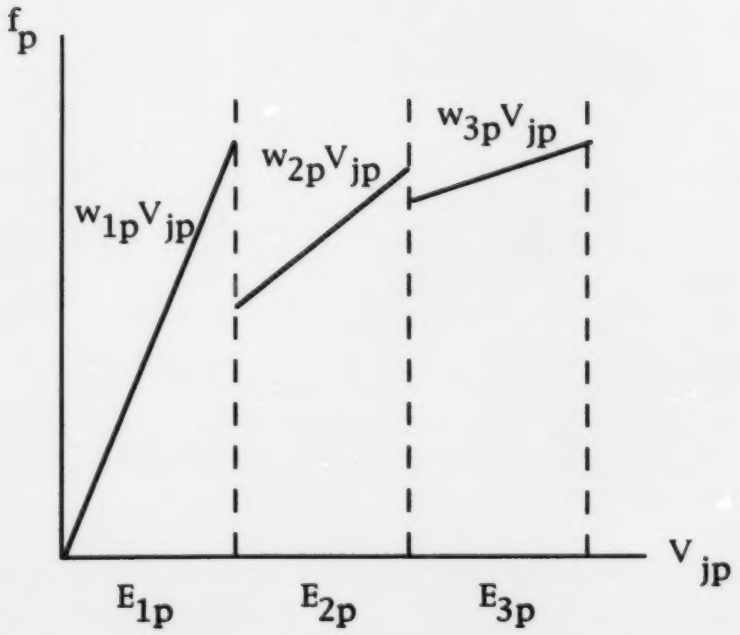

Figure 2 Non Monotonic function f_p

The testing results for the three experimental contexts are shown in Table I. It must be noted that in each case, we took the worst possible ranking for the vehicle selected by the rule or the expert. In other words, assuming that the non utility value of each vehicle is as follows:

vehicle 1: 0.1
vehicle 2: 0.1
vehicle 3: 0.1
vehicle 4: 0.2
vehicle 5: 0.4
vehicle 6: 0.4
vehicle 7: 0.5
vehicle 8: 0.6
vehicle 9: 0.8
vehicle 10: 0.8

and that vehicle 3 is selected by the expert, we then say that vehicle 3 is ranked third (and not first). This worst case approach was taken in order to avoid the ambiguity that would have occurred with a non-discriminating approach (a "method" that would assign the same non utility value to each vehicle should rank last, not first, the vehicle selected by the rule or the expert).

In the table, the first column shows, for each experiment, the number of dispatching situations for which the vehicle selected by the rule or the expert was ranked first by the linear programming model (over 100 dispatching situations). The second column shows the number of dispatching situations for which the selected vehicle was ranked first or second by the linear programming model. Finally, the third column looks at the three first ranks.

Experiment	Rank 1	Rank 1+2	Rank 1+2+3
1	78	100	100
2	86	100	100
3	67	85	91

Table I Ranking of the vehicles selected by the rule or the expert

As we can see from those figures, the vehicle selected by the dispatching rules was always ranked first or second by the linear programming model (cf. experiments 1 and 2). In the last experiment, the linear programming solution exactly reproduced 67 decisions of the expert. Moreover, the vehicle selected by the expert was ranked first or second in 85 dispatching situations, and ranked first, second or third in 91 dispatching situations out of 100.

Those results are interesting, particularly if we consider the fact that the vehicles that were ranked first and did not coincide with the expert's choice in the last experiment, were nevertheless vehicles that "make sense" according to that expert (i.e. no foolish choices).

We can now look at the weight patterns that were derived from the linear programming models (note that constant C in constraint (4) of model (3.2) was set to 100 in order to get the following weighting schemes).

In table II, we show the weights for experiments 1 and 2. Recall that only 3 attributes were used in these experiments, and that the attribute value domain was divided into 11 distinct intervals for each attribute.

Experiment	Attribute 1 (extra-mileage)	Attribute 2 (pick-up time)	Attribute 3 (delivery time)
1	$w_{i1} = 0.0$	$w_{i2} = 8.6$	$w_{i3} = 0.5$
2	$w_{i1} = 2.2$	$w_{i2} = 6.8$	$w_{i3} = 0.2$

* For any given attribute j, the weights w_{ij} are the same for all i, $1 \le i \le 11$

Table II Weighting Schemes for Experiments 1 and 2

The first decision rule uses attribute 2 (pick-up time) to filter the vehicles and then attribute 3 (delivery time) for the final selection. As we can see in Table II, the weights are all the same over each attribute domain, and are such that a large weight is assigned to attribute 2, and a smaller weight to attribute 3. As expected, the

weight for attribute 1 is zero, since it was not involved in the decision process.

For the second decision rule, attributes 2 (pick-up time) and 1 (extra-mileage) were used to filter the initial set of vehicles and attribute 3 (delivery time) was used for the final selection. Now, attribute 3 is involved in the decision process, and gets a positive weighting. As for the previous experiment, the weights are the same over each attribute domain. Hence, a simpler linear programming model with only one weight per attribute would have produced the same kind of results.

For the more complex experiment with the expert, we used 10 distinct attributes with 11 distinct intervals for each attribute domain. Since the pattern of weights is now highly non linear (that is, there are many distinct weight values for each attribute), we do not show here the detailed weighting sheme but rather emphasize the main trends. First, all the attributes have at least one non negative weight. The attributes with highest weigths are attribute 1 (extra-mileage) with 5 distinct weights varying from 0.6 to 1.7, attribute 4 (delay over 30 minutes for pick-up) with 3 distinct weights varying from 1.6 to 2.4 and attribute 5 (delay over 90 minutes for delivery) with three distinct weights between 1.3 and 2.1. The expert thus put much emphasis on extra-mileage and quality of service. We also observed larger weights for intervals with low values, and smaller weights for intervals with high values. Hence, the marginal increase gets smaller as the values get higher (like "concave" functions).

In order to enhance the predictive power of the model in the last experiment, we also evaluated the effect of putting less emphasis on the second component within the objective function of model (3.2). That component is aimed at minimizing the number of distinct intervals. We thus introduced the following parameter θ :

$$(3.2)\quad \text{Min}\ \sum_{(j,k)\in P} Z_{jk} + \theta \sum_{p}\sum_{i=1,\ldots,Ip-1} (Y^{-}_{ip} + Y^{+}_{ip})$$

and looked at the results for various θ values. Although decreasing the value of θ from 1 (as in the original experiment) to 0 did generate a weighting scheme with more distinct weights for each attribute, the overall predictive power of the model did not improve significantly by doing so.

The same observation did apply when we tried to increase the number of intervals E_{ip} for each attribute p (we used 11 intervals in the original experiment). The predictive power of the model remained the same for 11, 16 or 21 intervals. On the other hand, reducing that number from 11 to 3 did produce a substantial degradation in the predictive power of the model (only 38% of the vehicles selected by the expert were ranked first with 3 intervals, as compared to 67% with 11 intervals). We should recall that all values within an interval are considered to be "equivalent" and are replaced by the center of that interval in the linear programming model. If those intervals are too large, the vehicles all look more or less the same and the system cannot discriminate anymore between them.

The next section now concludes by presenting some comparisons between the global approach and the incremental weight learning approach.

6.4 Global Approach vs Incremental Approach

In this section, we compare the global approach, which is solving a single large linear programming problem, and the incremental weight learning approach, as described in section 5.

We did apply both methods to experiments 1 and 3, and the results are shown in Table III. As we can see in the table, the incremental scheme provides great benefits with respect both to computation time and the size of the problems to be solved. In experiment 1, only 38 out of the 900 preference constraints were added to the model, while in experiment 3 only 104 constraints were added. The speed up with respect to computation time is also quite substantial. Note that the total computation time for the incremental approach is the sum of the computation times for solving all the subproblems plus the time for checking and adding preference constraints to the linear program.

On the other hand, the total number of simplex iterations is not so different from one approach to the other (the total number of simplex iterations for the incremental approach is the sum of all the simplex iterations over all the subproblems). The fact that the incremental approach is solving small problems, while the global approach solves a large problem, accounts here for the speed up with respect to computation time.

Experiment 1	# Constraints	# Simplex Iterations	Computation time
Global	900	303	608 sec.
Incremental	38	178	216 sec.

Experiment 3	# Constraints	# Simplex Iterations	Computation time
Global	900	2367	1810 sec.
Incremental	104	2077	1050 sec.

Table III Global vs Incremental Approach

This incremental weight learning scheme is thus very promising, because of its potential for real-time applications. We can imagine a computer system that would be linked to an expert dispatcher working in a real dispatching environment. Each time a new decision is taken, the system uses that new decision to incrementally adjust itself to the dispatcher's decision process. That is, if the decision of the expert is coherent with the current set of weights, everyting is fine and the system stays idle. On the other hand, if the decision is not coherent with the current set of weights, the system adds to the model the preference constraints of the current request that are not satisfied, and solves the new model in order to get a new set of weights. Since each subproblem to be solved is small, building a (real-time) adaptive system along those guidelines look promising.

7. CONCLUDING REMARKS

The results presented above indicate that linear programming models are quite good at reproducing various dispatching behaviors. In that respect, the incremental weight learning scheme looks particularly attractive, because it is suitable for a real-time implementation. During the training phase, the computer system would be linked to an expert dispatcher taking decisions in real time, and would use each new decision to incrementally update its set of weights. After the training phase, the

computer system would then be able to use the learned weights to suggest vehicles to the dispatcher and thus act as a consultant system. This flexible learning approach would allow the same computerized system to be used in various dispatching environments. This is certainly a major asset as compared with the classical expert system approach where decision rules must be tailored to each application.

Finally, it is important to observe that the vehicle routing expert involved in the last experiment had to take decisions based on an *a priori* set of attributes. Since this expert was not a professional dispatcher, she accepted to take decisions based on such data. However, it is not clear if a professional dispatcher would have accepted our metrics for describing dispatching situations. It is thus an interesting problem in itself to determine (with the dispatcher) the measures that should be used so as to allow good decisions to be taken. Since no professional dispatcher was available, we had to "skip" this phase and a priori determine those metrics, to the best of our knowledge.

Acknowledgements. We would like to thank here the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Fonds pour la Formation de Chercheurs et l'Aide a la Recherche of Quebec Government (FCAR) for their financial support.

REFERENCES

[Beckwith and Lehmann 73] N.E. Beckwith and D.R. Lehmann, "The Importance of Differential Weights in Multiple Attribute Models of Consumer Attitude", Journal of Marketing Research 10, 141-145.

[Bodin et al. 83] L. Bodin, B. Golden, A. Assad and M. Ball, "Routing and Scheduling of Vehicles and Crews: The State of the Art", Computers and Operations Research 10(2), 63-211.

[Green and Rao 71] P.E. Green and V.R. Rao, "Conjoint Measurement for Qualifying Judgement Data", Journal of Marketing Research 8, 355-363.

[Horsky and Rao 84] D. Horsky and M.R. Rao, "Estimation of Attribute Weights from Preference Comparisons", Management Science 30, 801-822.

[Jacquet-Lagreze and Siskos 82] E. Jacquet-Lagreze and J. Siskos, "Assessing a set of additive utility functions for multicriteria decision

making: the UTA method", European Journal of Operational Research 10, 151-164.

[Jain et al. 79] A.K. Jain, F. Acito, N.K. Malhotra and V. Mahojan, "A Comparison of the Internal Validity of Alternative Parameter Estimation Methods in Decompositional Multi-Attribute Preference Models", Journal of Marketing Research 16, 313-322.

[Kain et al. 79] A.K. Jain, F. Acito, N.K. Malhotra and V. Mahajan, "A Comparison of the Internal Validity of Alternative Parameter Estimation Methods in Decomposional Multi-attribute Preference Models", Journal of Marketing Research 16, 313-322.

[Lynch 85] J.G. Lynch, "Uniqueness Issues in the Decompositional Modeling of Multiattribute Overall Evaluations: an Information Integration Perspective", Journal of Marketing Research 22, 1-19.

[Marsten 87] R. Marsten, "XLP Technical Reference Manual", XMP Software Inc..

[Mustafi and Xavier 85] C.K. Mustafi and M.J. Xavier, "Mixed Integer Linear Programming Formulation of a Multi-attribute Threshold Model of Choice", Journal of the Operational Research Society 36, 935-942.

[Oral and Kettani 89] M. Oral and O. Kettani, "Modelling the Process of Multiattribute Choice", Journal of the Operational Research Society 40(3),281-291.

[Psaraftis 80] H. Psaraftis, "A Dynamic Programming Solution to the Single Vehicle Many-to-Many Immediate Request Dial-a-Ride Problem", Transportation Science 2, 130-154.

[Quinlan 86] Quinlan J.R., "Induction of Decision Trees", Machine Learning 1(1), 81-106.

[Rousseau and Roy 88] J.M. Rousseau and S. Roy, "Répartition Assistée par Ordinateur: la Description d'un Prototype", Publication du Centre de Recherche sur les Transports #564, Université de Montréal.

[Rumelhart and McClelland 86] D. Rumelhart and J.L. McClelland, Parallel Distributed Processing: Explorations in the MicroStructure of Cognition, volumes 1 and 2, The MIT Press.

[Shocker and Srinivasan 79] A.D. Shocker and V. Srinivasan, "Multiattribute approaches for Product Concept Evaluation and

Generation: A critical Review", Journal of Marketing Research 16, 159-180.

[Srinivasan and Shocker 73a] V. Srinivasan and A.D. Shocker, "Linear Programming Techniques for Multidimensional Analysis of Preferences", Psychometrica 38, 337-369.

[Srinivasan and Shocker 73b] V. Srinivasan and A.D. Shocker, "Estimating the Weights of Multiple Attributes in a Composite Criterion using Pairwise Judgements", Psychometrica 38, 473-493.

[Wilson and Colvin 79] N. Wilson, N. Colvin, "Computer Control of the Rochester Dial-a-Ride System", CTS Report Number 77-22, Cambridge, Mass., MIT, Center for Transportation Studies.

[Wittink and Catin 81] D.R. Wittink and P. Catin, "Alternative Estimation Methods for Conjoint Analysis: A Monte-Carlo Study", Journal of Marketing Research 18, 101-106 .